HARDWARE-PRAKTIKUM

Versuch L-4

Komplexe Schaltwerke

Fachbereich Informatik

Universität Kaiserslautern

Versuch L-4

In diesem Versuch soll ein Rechenwerk zur Multiplikation von zwei vorzeichenlosen 4 Bit Dualzahlen X und Y nach der Methode der Seriell-Parallelen Multiplikation entworfen und aufgebaut werden. Das 8 Bit große Produkt P soll auf einer Anzeige dargestellt werden. Die Eingabe für X und Y erfolgt über Schalter.

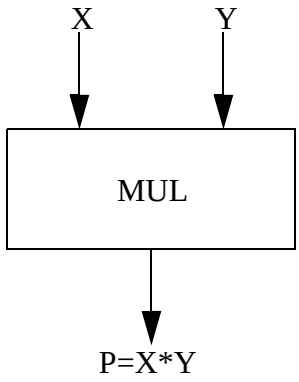


Abbildung 1

Die Schaltung läßt sich in ein Steuerwerk und ein Rechenwerk unterteilen.

Das Rechenwerk

Der Aufbau eines Rechenwerkes zur Multiplikation soll anhand eines 4 Bit x 4 Bit Multiplizierers schrittweise demonstriert werden.
Die Multiplikation zweier vorzeichenloser Dualzahlen erfolgt schriftlich analog zu der uns vertrauten Multiplikation zweier Dezimalzahlen. Man bildet Teilprodukte, die stellenrichtig aufaddiert werden müssen.
Beispiel:

$X = X_3 \cdot 2^3 + X_2 \cdot 2^2 + X_1 \cdot 2^1 + X_0$, n = 4

$Y = Y_3 \cdot 2^3 + Y_2 \cdot 2^2 + Y_1 \cdot 2^1 + Y_0$, n = 4

$X = 12 = 1100_2$, $Y = 6 = 0110_2$, $12 \cdot 6 = 72 = 1001000_2$

```
1100 * 0110
     0000000
      110000
       11000
+       0000
      1001000
```

Vereinfachend kommt im Zweiersystem hinzu, daß die Y_i des Multiplikators Y nur den Wert 0 oder 1 haben können. Damit ist zum Aufaddieren der Multiplikand für $Y_i = 1$ stellenverschoben zu addieren oder für $Y_i = 0$ eine Addition zu unterlassen.

Bild 2 zeigt einen einfachen Multiplizierer, der die Multiplikation $X \cdot Y_i$ parallel, die Additionen der Teilprodukte aber seriell vornimmt (Seriell-Parallele Multiplikation)

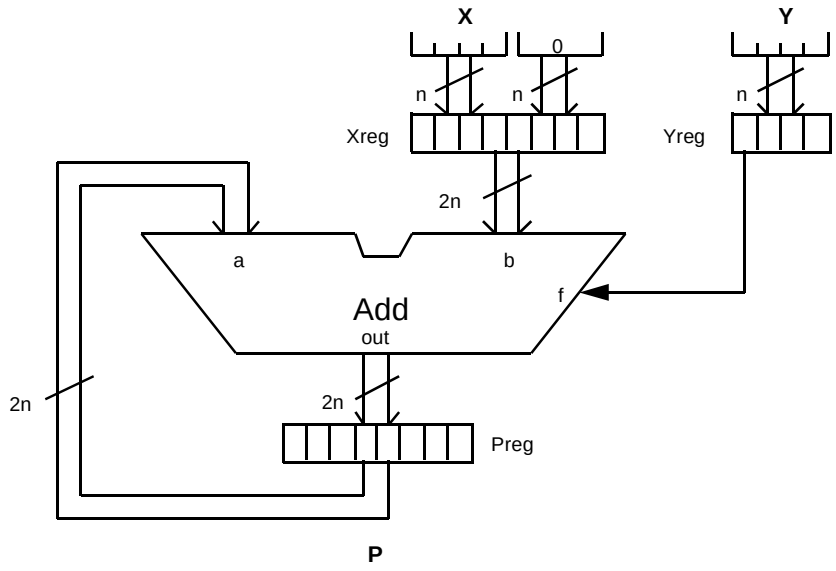


Abbildung 2

X_{reg} ist ein nach rechts und Y_{reg} ein nach links schiebendes Schieberegister, das parallel geladen werden kann. Preg ist das Ergebnisregister. Im Bild ist n = 4, d.h. das Ergebnis hat maximal 8 Bit Breite.

Die Schaltung funktioniert so:

lade X_{reg} mit $X \cdot 2^n$, lade Y_{reg} mit Y, lade P mit 0
for i = 1 to n do
    shift X_{reg} right
lade P_{reg} mit Ergebnis aus Addierer
shift Y_{reg} left

Der Addierer liefert für f = 0 am Ausgang a, für f = 1 am Ausgang a+b. Er muß eine Breite von 2n haben. Die Berechnung der Teilprodukte berücksichtigt zuerst das MSB des Multiplikators und zuletzt das LSB.

P enthält damit nacheinander die Werte

$$0,\ X \cdot Y_{n-1} \cdot 2^{n-1}\ ,\ X \cdot Y_{n-1} \cdot 2^{n-1} + X \cdot Y_{n-2} \cdot 2^{n-2},\ ...$$

$$..,\ X \cdot Y_{n-1} \cdot 2^{n-1} + X \cdot Y_{n-2} \cdot 2^{n-2} + \ldots + X \cdot Y_1 \cdot 2^1 + X \cdot Y_0 \cdot 2^0 = X \cdot Y$$

Eine Vereinfachung wird in Bild 3 gezeigt.

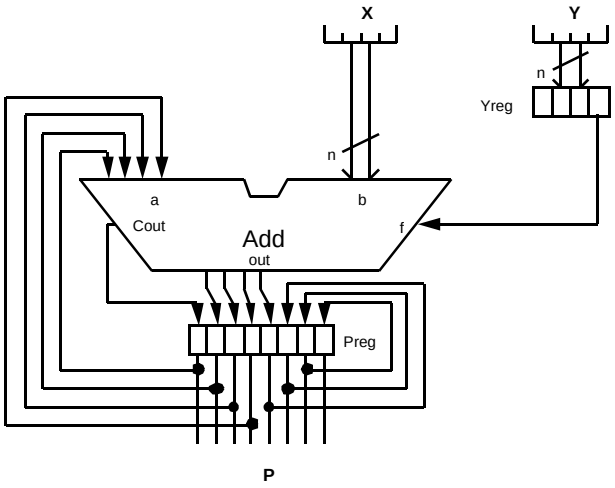


Abbildung 3

Hier ist Y_{reg} ein nach rechts schiebendes parallel ladbares Schieberegister und P das Ergebnisregister. Der Addierer hat eine Breite von n Bits (hier n=4).

Die Berechnung berücksichtigt nun zuerst das LSB des Multiplikators und zuletzt das MSB:

lade Y_{reg} mit Y, lade P_{reg} mit 0
for i=1 to n do
    lade P_{reg} mit Ergebnis aus Addierer
    Shift Y_{reg}

Das stellenrichtige Aufaddieren wird hier durch die Verdrahtung von P_{reg} erreicht. Diese bewirkt ein Schieben nach rechts, was im Zweiersystem einer Division durch 10 (dezimal 2) entspricht.

P_{reg} enthält enthält nacheinander die Werte

$$0,\ (0 + X \cdot Y_0 \cdot 2^n)/2,\ ((0 + X \cdot Y_0 \cdot 2^n)/2 + X \cdot Y_1 \cdot 2^n)/2,\ \ldots$$

$$\ldots,\ ((\quad) + X \cdot Y_{n-1} \cdot 2^n)/2 = X \cdot Y_0 \cdot 2^0 + \ldots + X \cdot Y_{n-1} \cdot 2^{n-1} = X \cdot Y$$

Auch das Y_{reg} läßt sich noch einsparen. Bei obiger Schaltung fällt auf, daß die niederwertige Hälfte von P_{reg} erst nach und nach mit signifikanten Daten gefüllt wird. Andererseits benötigt

man von Y nur die Y_i, für die noch keine Multiplikation durchgeführt wurde. Man kann, wie in Bild 4 zu sehen, Y mit in P_{reg} ablegen:

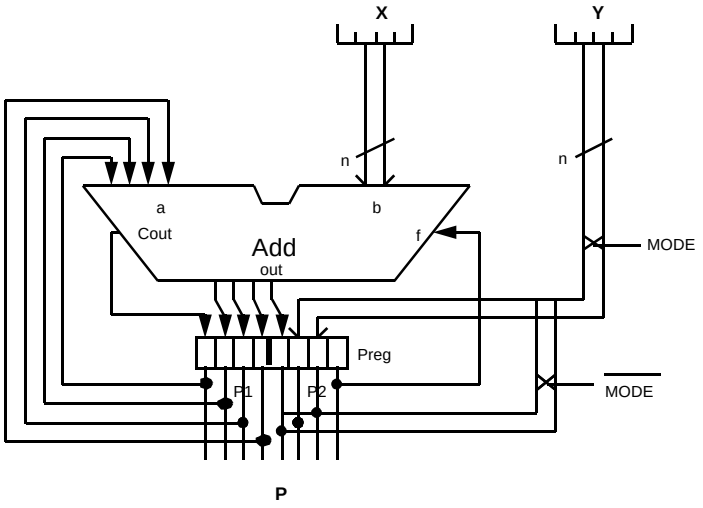


Abbildung 4

Der Addierer hat wiederum eine Breite von n Bit. P_{reg} ist ein parallel ladbares Register, das infolge der äußeren Verdrahtung als nach rechts schiebendes Schieberegister funktioniert:

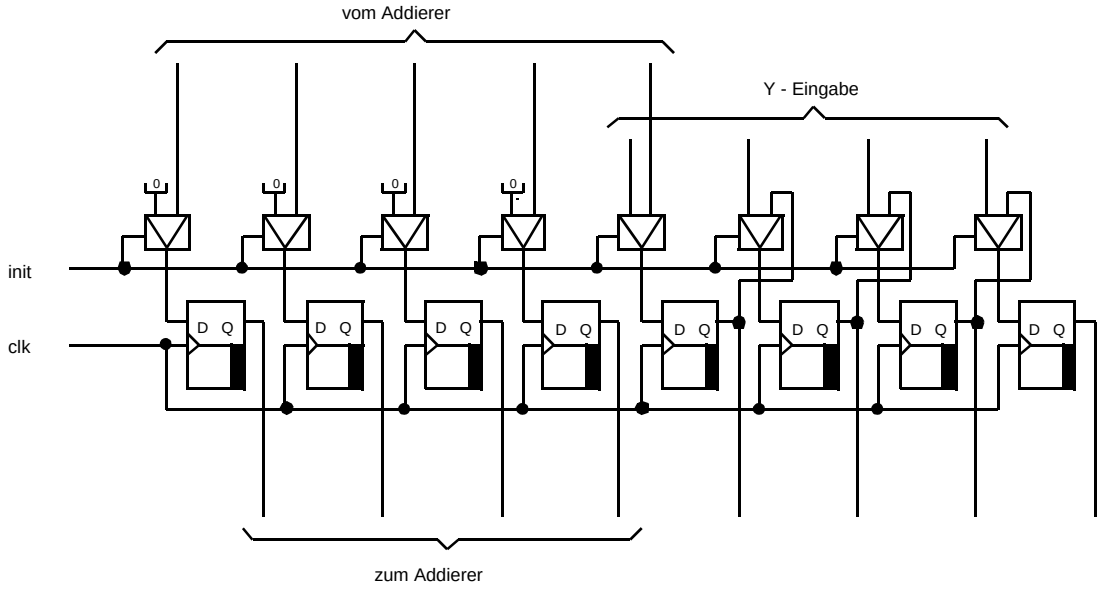


Abbildung 5

Das Register P_{reg} ist in zwei Hälften P_1 und P_2 mit n Bit aufgeteilt.
Die Schaltung arbeitet jetzt so:

lade P_1 mit 0, lade P_2 mit Y
for i= 1 to 4
lade P mit Ergebnis aus Addierer und schiebe intern

Nach n Schritten liegt das Produkt in P_{reg} vor.

Literatur

W. Giloi, H. Liebig
Logischer Entwurf digitaler Systeme
Springer Verlag Berlin Heidelberg New York

Rolf Hoffmann
Rechenwerke und Mikroprogrammierung
R. Oldenbourg Verlag München Wien, Reihe Datenverarbeitung

K. Waldschmitt
Schaltungen der Datenverarbeitung
B.G. Teubner Stuttgart

Aufgabenstellung:

Entwerfen Sie einen 4 Bit x 4 Bit Multiplizierer. Verwenden Sie die in Bild 4 dargestellte optimierte Schaltungsvariante für das Rechenwerk.
Die Eingabewerte für X und Y kommen von einer Schalterreihe. Verwenden Sie folgende Zuordnung:

Eingabe	Anschluß	Eingabe	Anschluß
X0	S0	Y0	S4
X1	S1	Y1	S5
X2	S2	Y2	S6
X3	S3	Y3	S7

Den Wert des P_{reg}s stellen Sie auf einer Hexadezimalanzeige dar. Ihr Einschub enthält eine sechsstellige Anzeigeeinheit mit folgendem Aufbau:

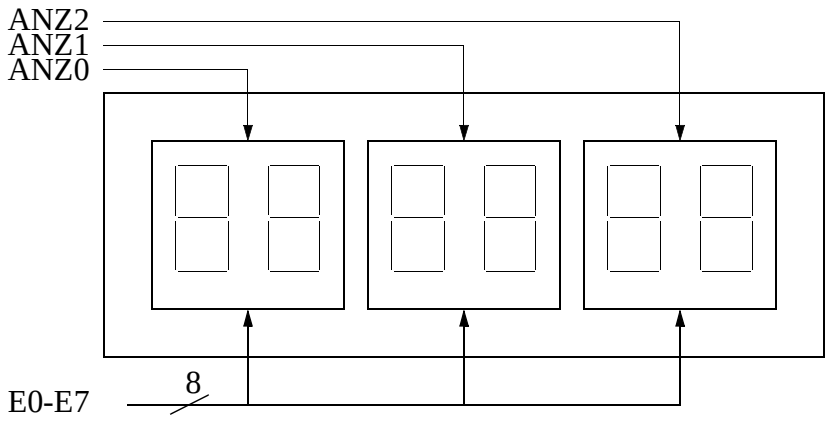


Abbildung 6

Zur Eingabe in die Anzeigeneinheit steht ein 8-Bit-Eingabewort E0-E7 zur Verfügung. Die Auswahl, welche der drei Anzeigepaare das gerade anliegende Eingabewort übernehmen soll, erfolgt durch die drei Steuerleitungen ANZ0, ANZ1 und ANZ2. Solange eine dieser Steuerleitungen auf "LOW" gesetzt bleibt, wird das Eigabewort in die zugehörige Anzeige übernom-

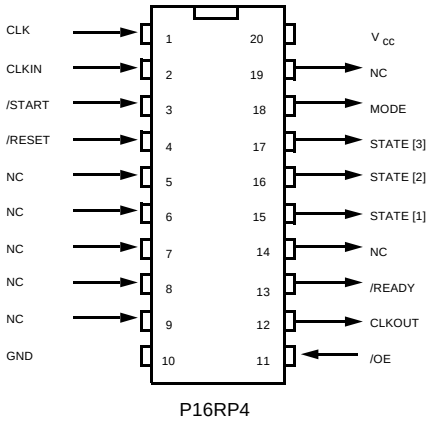
men. Da die Speicherregister der Anzeigeneinheit pegelgesteuert sind, muß das Eingabewort mindestens noch 50ns stabil an den Eingängen anliegen, nachdem die Steuerleitung auf "HIGH" gelegt worden ist.
Verwenden Sie zur Darstellung des Ergebnisses die rechts liegenden zwei Anzeigestellen. Übernehmen Sie das Ergebnis durch Aktivieren von ANZ2 erst, wenn die Berechnung abgeschlossen ist.
Die Belegung der Anschlüsse auf den Protoboard entnehmen Sie Tabelle I. Unterteilen Sie Ihren Entwurf in ein Rechen- und ein Steuerwerk.
Das Rechenwerk erhält seine Eingabe X und Y von der Schalterreihe. Zur Verfügung steht ein 4 Bit Volladdierer vom Typ 74LS283. Da dieser immer am Ausgang die Summe seiner Eingangswerte liefert, müssen Sie den Addierer aus Bild 4 durch einen Addierer mit einem nachgeschalteten Multiplexer ersetzen, der wahlweise das Ergebnis der Addition oder den Eingang a des Addierers durchschaltet. Damit wird die Funktion der Eingangsleitung f nachgebildet. Das Register P_{reg} kann zusammen mit dem zusätzlichen Multiplexer in einem PAL vom Typ 20RP8 realisiert werden. Für das Rechenwerk sind somit nur zwei ICs erforderlich. Die Belegung des PALs entnehmen Sie Tabelle 2.
Das Steuerwerk soll als Automat realisiert werden. Nach Aktivieren des Tasters T (act. low) auf der Anzeigeeinheit soll die Berechnung gestartet werden. Nach Vorliegen des Ergebnisses soll das Ergebnis bis zum erneuten Starten der Berechnung in Preg gültig bleiben. Für die Anzeigeeinheit ist ein Übernahmeimpuls zu erzeugen. Zum Aufbau des Steuerwerkes genügt ein PAL vom Typ 16RP4. Die Belegung entnehmen Sie bitte Tabelle 3. Beschreiben Sie das Steuerwerk durch ein Zustandsdiagramm. Entwerfen Sie die PLPL-Programme zur Beschreibung der PALs.
Geben Sie die PLPL-Programme ein und programmieren Sie die zwei benötigten GALs.
Bauen Sie die Schaltung auf dem Protoboard auf und testen Sie mit vorentworfenen Beispielen.
Zeigen Sie den Ablauf einer Berechnung mit Hilfe des Logikanalysators. Stellen Sie dazu den Inhalt von P_{reg} und den Zustand des Steuerwerks mit den Ausgaben dar!

Tabelle 1: Hexadezimale Multiplikation

	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
2	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E
3	03	06	09	0C	0F	12	15	18	1B	1E	21	24	27	2A	2D
4	04	08	0C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	05	0A	0F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	06	0C	12	18	1E	24	2A	30	36	3C	41	48	4E	54	5A
7	07	0E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	08	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	09	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	0B	16	21	2C	37	41	4D	58	63	6E	79	84	8F	9A	A5
C	0C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

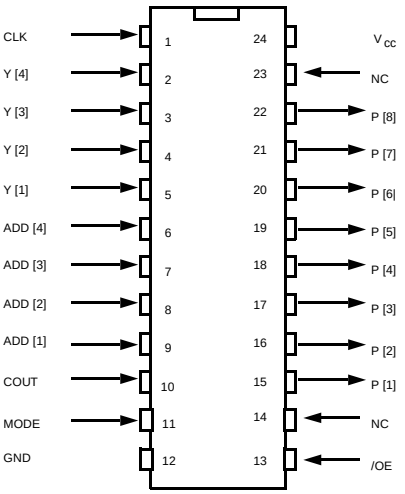
Anschlußbelegung des Protoboards

	Belegung	
A0	Schalter	S0
A1		S1
A2		S2
A3		S3
A4		S4
A5		S5
A6		S6
A7		S7
B0	Eingabewort	E0 der Hexidezimalanzeige
B1		E1
B2		E2
B3		E3
B4		E4
B5		E5
B6		E6
B7		E7
C0	Steuerleit.	ANZ0
C1		ANZ1
C2		ANZ2
C3	Taster	T (act. low)
C4		
C5		
C6		
C7		
D0	Spalteneing.	SE0
D1		SE1
D2		SE2
D3		SE3
D4	Zeilenausgang	ZA0
D5		ZA1
D6		ZA2
D7		ZA3
E0		
E1		
E2		
E3		
E4		
E5		
E6	Einzeltakt	
E7	Reset (act. low)	
F0		
F1		
F2		
F3		
F4		
F5		
F6		
F7		
G0		
G1		



P16RP4

Tabelle 2



P20RP8

Tabelle 3

Fragen L-4

1. Grundlagen - Fragen über Schaltwerke, Bausteine, siehe L-1, L-2, L-3
2. Was ist die «binäre« Zahlendarstellung? Wie können Zahlen mit Vorzeichen dargestellt werden?
3. Wie funktioniert die binäre Addition/Subtraktion (auf dem Papier)? Was geschieht bei negativen Zahlen? Warum entsteht ein Überlauf?
4. Wie funktioniert die binäre Multiplikation (auf dem Papier)? Was geschieht bei Zahlen mit Vorzeichen? Kann ein Überlauf entstehen? Wie groß ist der Definitionsbereich (der Faktoren)?
5. Wie realisiert man einen Binäraddierer? Was ist ein Halbaddierer/Volladdierer?
6. Warum beginnt die aufgebaute Schaltung mit dem LSB zu multiplizieren? Gibt es auch eine Variante, die mit dem MSB beginnt?
7. Aus welchen logischen Komponenten besteht Ihr Rechenwerk? Wie teilen sich diese auf die physikalischen Bausteine auf?
8. Wie sieht das Zustandsdiagramm des Steuerwerks aus? Was sind die Ein- und Ausgänge? Wieviele Zustände sind mindestens erforderlich?
9. Welche Grundstruktur realisiert das Steuerwerk?
10. Im Versuch wird der Takt des Rechenwerks im Steuerwerk erzeugt. Welche Gefahr besteht dabei? Was wäre eine sichere Lösung?
11. Wie funktioniert ein Schieberegister? Wie unterscheidet sich das Schieberegister im Rechenwerk von einem konventionellen Schieberegister?